An Efficient Transformation for Klee's Measure Problem in the Streaming Model

* * † † †

Abstract

Given a stream of rectangles over a discrete space, we consider the problem of computing the total number of distinct points covered by the rectangles. This can be seen as the discrete version of the two-dimensional Klee's measure problem for streaming inputs. We provide an (ϵ, δ)-approximation for fat rectangles. For the case of arbitrary rectangles, we provide an $\mathcal{O}(\sqrt{\log U})$-approximation, where U is the total number of discrete points in the two-dimensional space. The time to process each rectangle, the total required space, and the time to answer a query for the total area are polylogarithmic in U. Our approximations are based on an efficient transformation technique which projects rectangle areas to one-dimensional ranges, and then uses a streaming algorithm for the Klee's measure problem in the one-dimensional space. The projection is deterministic and to our knowledge it is the first approach of this kind which provides efficiency and accuracy trade-offs in the streaming model.

1 Introduction

The well-known two-dimensional *Klee's measure problem* (KMP) [7] can be stated as follows: given a collection of m axis-aligned rectangles, how quickly can we compute the area of their union? This problem has been studied extensively in the literature [3, 6, 12] with the best known bounds of $\mathcal{O}(m \log m)$ and $\mathcal{O}(m)$ for the time and space requirements for an exact answer for m rectangles.

In this paper, we consider the problem of estimating the discrete version of the classical two-dimensional KMP in the streaming model. In this case, the data stream consists of rectangular elements over a discrete two-dimensional grid of points, and the task is to efficiently estimate at any time the number of distinct grid points occupied by the rectangles that have arrived so far. Following the literature, we hereafter denote this problem of finding the number of distinct elements by F_0 (referred to as the *zeroth frequency moment*) [1, 10, 11].

The motivation to study KMP in the streaming model is due to spatial and temporal data that arise in many domains such as VLSI layout processing and sensor networks. A spatial database, e.g. OpenGIS[1], deals with a large collection of relatively simple geometric objects, of which rectangles are the most basic types. Moreover, query processing in the constraint database model [8] can also be seen as a computation over the set of geometric objects, e.g. [2]. The streaming setting makes sense in online scenarios of the aforementioned applications when the workspace is very limited such that rescanning the entire dataset is not feasible.

A recent work in the F_0 of KMP in the streaming model is due to Tirthapura and Woodruff [11], who gave an (ϵ, δ)-approximation algorithm with space as well as processing time per rectangle $(\frac{1}{\epsilon}\log(\frac{mU}{\delta}))^{\mathcal{O}(1)}$, $0 < \epsilon, \delta < 1$, where U is the total number of grid points in the two-dimensional space[2]. Comparing to their algorithm, the algorithm we present here is very simple, does not depend on m, and exhibits different tradeoffs.

Contributions. We consider a $2^n \times 2^n$ two-dimensional grid with $U = 2^{2n}$ points. The input stream consists of a set of m elements $\Upsilon = \{x_0, x_1, \ldots, x_{m-1}\}$, where each x_i is an $a_i \times b_i, 0 \leq a_i, b_i < \sqrt{U}$, rectangle of discrete points, and x_{m-1} is the last element that has arrived so far. Let A denote the total area (number of distinct discrete points) of the rectangles in Υ which have arrived so far in the stream. We present and analyze an algorithm that returns an estimate $\text{est}(A)$ of A.

Our first result is for "fat" rectangles − we say that a rectangle is *fat* if the ratio g of its side lengths (i.e., the aspect ratio) is between $\frac{1}{c} < g < c$, where $c \geq 1$ is some constant, e.g. [4]. We give an algorithm which provides an (ϵ, δ)-approximation of A (that is, F_0). Given $0 < \epsilon, \delta < 1$, an approximation algorithm is said to (ϵ, δ)-approximate F_0 if the estimated output $\hat{F}_0$ satisfies $\mathbf{Pr}[|\hat{F}_0 - F_0| < \epsilon F_0] > 1 - \delta$. Moreover, our streaming approximation algorithm achieves the following time and space complexities for fat rectangles: (i) the amortized processing time per rectangle is $\mathcal{O}(\frac{1}{\epsilon}\log\frac{U}{\epsilon}\log\frac{1}{\delta})$; (ii) the workspace needed is $\mathcal{O}(\frac{1}{\epsilon^2}\log U \log\frac{1}{\delta})$ bits; and

(iii) the time to answer a query for F_0 is $\mathcal{O}(1)$.

For the general case of arbitrary rectangles (the rectangles with any ratio of side lengths), we present a streaming algorithm which provides an $\mathcal{O}(\sqrt{\log U})$-approximation of A, such that $\Omega(1/\sqrt{\log U}) \leq \text{est}(A)/A \leq \mathcal{O}(\sqrt{\log U})$; the approximation bound holds with constant probability. Moreover, it ensures that: (i) the amortized processing time per rectangle is $\mathcal{O}(\log U \cdot \log\log U)$; (ii) the workspace needed $\mathcal{O}(\log^2 U \cdot \log\log U)$ bits; and (iii) the time to answer a F_0 query is $\mathcal{O}(\log U)$.

The main idea is to transform each input rectangle to an interval (range), such that the estimate of F_0 for the intervals provides an estimate for A. Our two-dimensional approximation is based on the *range-efficient*[3] (ϵ, δ)-approximation algorithm of [10].

Our algorithm implements an efficient proximity transformation technique of rectangles to ranges based on a Z-ordering [9] (note that a depth-first traversal of a quadtree is essentially a Z-ordering). The proximity-based transformation is deterministic and partitions the data stream into buckets according to the aspect ratio of the rectangles. In the general case we use $\mathcal{O}(\log U)$ buckets, while for the case of fat rectangles we only use one bucket. We then apply a range-efficient algorithm for each bucket instance independently. The algorithm requires first to normalize a rectangle and then project it to a range. The normalization helps to preserve the intersection properties of the rectangles even when they are transformed to ranges, which further helps to obtain good approximations. In the analysis, we bound the error due to normalization and also due to the projection on ranges.

To the best of our knowledge, this is the first transformation algorithm for KMP that improves significantly on the previous solutions for fat rectangles [4, 11].

Related Work. For the classical (non-streaming) KMP, Bentley [3] described a deterministic time-optimal $\mathcal{O}(m \log m)$ time and $\mathcal{O}(m)$ workspace solution. This solution is based on reducing the problem to m one-dimensional KMPs [7] by sweeping a vertical line across the area. Some recent work tried to minimize the space and time requirements for the efficient computation of the area of the union, e.g. [6, 12]. Particularly, Chen and Chan [6] gave an algorithm that runs in $\mathcal{O}(m^{3/2} \log m)$ time but needs $\mathcal{O}(\sqrt{m})$ extra workspace. Vahrenhold [12] minimizes the extra space to $\mathcal{O}(1)$ with the same running time. All the aforementioned solutions for KMP are deterministic and compute the *exact* area. Moreover, no deterministic algorithm has improved the best known bounds of $\mathcal{O}(m \log m)$ and $\mathcal{O}(m)$ for the time and space. Therefore, recent focus has been on approximation algorithms. To this end, Bringmann and Friedrich [5] gave a $(1 \pm \epsilon)$-approximation algorithm for any $0 < \epsilon < 1$. However, it has space complexity that is still linear in the size of the input.

A difficulty in obtaining tighter bounds on time and space complexities of KMP stems from the fact that [3, 6, 12] use explicit sorting algorithms and tree-based data structures (e.g. quadtrees) to handle rectangles, where such data structures need at least $\Omega(\log m)$ time to process an individual element using $\mathcal{O}(m)$ space. A natural question that arises is whether tighter bounds on time and space requirements are achievable. According to the literature, computing F_0 *exactly* requires space linear in the number of distinct values [1]. Therefore, we opt to design streaming approximation algorithms for KMP that require very limited space.

Outline of Paper. We give a KMP streaming algorithm in Section 2 and discuss the normalization in Section 3. In Section 4, we give an algorithm to transform a rectangle to a normalized approximation based on a Z-ordering. We analyze the transformation algorithm for the one bucket case (fat rectangles) in Section 5 and do the same for many buckets (general case) in Section 6. We conclude the paper in Section 7. Proofs are deferred to the full version due to space limitations.

2 KMP Streaming Algorithm

Algorithm 1 is an approach for estimating the total number of distinct points A covered by a streaming set Υ of m rectangles in $\mathbb{Z}_n^2$. The basic idea is to transform each rectangle to one or more one-dimensional ranges and then use a range-efficient algorithm to estimate the number of discrete points used by the stream of ranges. The challenge is to perform the transformation in such a way that the estimate from the ranges is a good approximation of A. In order to achieve good approximations, we first *normalize* the rectangles, by aligning them into appropriate space points whose coordinates are multiples of 2. We then separate the rectangles into different buckets according to their normalization. Each bucket has range mapping characteristics which help to accurately estimate the respective covered areas. We run a range-efficient algorithm to each bucket, which we combine to obtain the resulting estimate for A.

Algorithm 1 initializes χ buckets $\mathbb{B}_0, \ldots, \mathbb{B}_{\chi-1}$ of normalized rectangles in $\mathbb{Z}_n^2$. Each rectangle $x \in \Upsilon$ is transformed to one or more normalized rectangles $y'_1, y'_2, \ldots$ whose union is an approximation of x. Then, each y'_j is inserted into some appropriate bucket $\mathbb{B}_{i_j}$. Each $y'_j \in \mathbb{B}_{i_j}$ is immediately mapped to a one-dimensional range using a projection function appropriate for $\mathbb{B}_{i_j}$. In this way, bucket $\mathbb{B}_{i_j}$ produces a stream of ranges. Some known range-efficient (ϵ, δ)-approximation algorithm is applied to the stream of ranges to find an es-

[3]

Algorithm 1: A rectangle area estimation algorithm

Input: A streaming set Υ of m rectangles in $\mathbb{Z}_n^2$;
Output: An estimate $\text{est}(A)$ of total number of distinct discrete points A covered by the rectangles in Υ;

1 **Initialization:**
2 Define χ buckets $\mathbb{B}_0, \ldots, \mathbb{B}_{\chi-1}$ of normal rectangles in $\mathbb{Z}_n^2$ (initially the buckets are empty);
3 **When a new rectangle $x \in \Upsilon$ arrives:**
4 *Normalization:* Transform x into a sequence of normal rectangles $y'_1, y'_2, \ldots$, and then assign each y'_j to some appropriate bucket $\mathbb{B}_{i_j}$;
5 *One-dimensional mapping:* Map each y'_j to a range r_j using an appropriate projection based on $\mathbb{B}_{i_j}$;
6 Apply a range-efficient (ϵ, δ)-approximation algorithm to update with respect to r_j an estimate of A'_{i_j}, the total area (total discrete points) of the normal rectangles in $\mathbb{B}_{i_j}$;
7 Maintain $E_{\max} = \max_i \text{est}(A'_i)$ and $E_{\text{sum}} = \sum_{i=0}^{\chi-1} \text{est}(A'_i)$; the maximum and the sum among the χ bucket estimates;
8 **When an estimate for F_0 is asked for:**
9 Return $\text{est}(A) = \sqrt{E_{\max} E_{\text{sum}}}$;

timate of A'_i, the total number of discrete points occupied by the normalized rectangles in $\mathbb{B}_i$. Algorithm 1 then maintains $E_{\max}$ and E_{sum}, the maximum and the sum among all the bucket estimates, respectively. When an estimate $\text{est}(A)$ is asked for, the algorithm returns $\text{est}(A) = \sqrt{E_{\max} E_{\text{sum}}}$.

One can apply any range-efficient (ϵ, δ)-approximation algorithm for F_0 on each bucket $\mathbb{B}_i$ (step 6). Here, we use the Hits algorithm of [10]. It has the following time and space complexities for F_0 for each bucket $\mathbb{B}_i$, where $U = 2^{2n}$.

Theorem 1 (Pavan and Tirthapura [10]) *Given $0 < \epsilon < 1$ and $0 < \delta < 1$, algorithm* Hits *(ϵ, δ)-approximates F_0 with space complexity $\mathcal{O}(\frac{1}{\epsilon^2} \log U \log \frac{1}{\delta})$ bits, amortized time taken to process a range $\mathcal{O}(\log \frac{U}{\epsilon} \log \frac{1}{\delta})$, and time taken to process a query for F_0 at any time $\mathcal{O}(1)$.*

In section 4 we give a transformation Proximity of rectangles to ranges which will enable us to provide two versions of Algorithm 1, one with a single bucket (fat rectangles), and the other with multiple buckets (arbitrary rectangles). We continue with first describing the normalization that we use.

3 Normalization

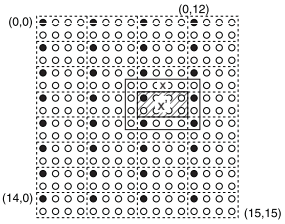
We first start with some basic definitions for ranges and their normalizations, and then we extend the definitions for normalized rectangles.

Ranges. For integer $n \geq 0$, let $\mathbb{Z}_n = \{0, 1, \ldots, 2^n - 1\} \subset \mathbb{Z}$ be a one-dimensional space of 2^n discrete integer points. A range (or interval) $r = [p_1, p_2]$, where $0 \leq p_1 \leq p_2 < 2^n$, consists of all the points between p_1 and p_2. Denote with $|r| = p_2 - p_1 + 1$ the size of range r which is the number of points in it.

The *α-normal subset* of $\mathbb{Z}_n$, denoted $\mathbb{W}_n^\alpha$, for integer $0 \leq \alpha \leq n$, consists of every $(2^\alpha)^{\text{th}}$ element of $\mathbb{Z}_n$, namely, $\mathbb{W}_n^\alpha = \{p \in \mathbb{Z}_n \ : \ p = i2^\alpha \wedge i \in \mathbb{Z}\}$. We will refer to the elements of $\mathbb{W}_n^\alpha$ as *normal points*. The normal subset $\mathbb{W}_n^\alpha$ induces $2^{n-\alpha}$ *normal ranges* of size 2^a such that each starts at a normal point. We will also use the notation $\mathbb{W}_n^\alpha$ to denote the normal ranges. Let $\mathbb{W}_n = \bigcup_{\alpha=0}^{n-1} \mathbb{W}_n^\alpha$ denote the set of all possible normal ranges (and respective normal points).

Rectangles. All the definitions for one-dimensional space $\mathbb{Z}_n$ extend to the two-dimensional space $\mathbb{Z}_n^2 = \mathbb{Z}_n \times \mathbb{Z}_n$ of discrete integer points. Space $\mathbb{Z}_n^2$ can be viewed as an array of points such that for any point $(p, q) \in \mathbb{Z}_n^2$, p corresponds to a row and q to a column (there are 2^n rows and 2^n columns). The upper left corner of $\mathbb{Z}_n^2$ is point $(0, 0)$, and the lower right corner is point $(2^n - 1, 2^n - 1)$. (See the figure below.)

A rectangle $x = \langle (p_1, q_1), (p_2, q_2) \rangle$ is a subset of $\mathbb{Z}_n^2$, where $p_1, p_2, q_1, q_2 \in \mathbb{Z}_n$ with $p_1 \leq p_2$ and $q_1 \leq q_2$, such that x contains all points $\{(p, q) \ : \ p_1 \leq p \leq p_2 \text{ and } q_1 \leq q \leq q_2\}$. Note that (p_1, q_1) is the north-west corner of x, while (p_2, q_2) is the south-east corner. We say that x is a $a \times b$ rectangle with side lengths $a = p_2 - p_1 + 1$ and $b = q_2 - q_1 + 1$. We denote with $|x| = a \cdot b$ the size of rectangle x which is the number of points in it.



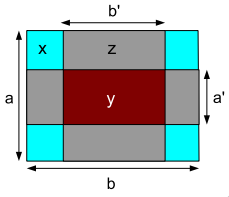
For any integers $0 \leq \alpha, \beta \leq n$, define the *$(\alpha, \beta)$-normal subset* $\mathbb{W}_n^{\alpha,\beta} = \mathbb{W}_n^\alpha \times \mathbb{W}_n^\beta \subseteq \mathbb{Z}_n^2$. Each element $(p, q) \in \mathbb{W}_n^{\alpha,\beta}$ is a *normal point* for which it holds $p = i2^\alpha$ and $q = j2^\beta$, for integers i and j. In other words, set $\mathbb{W}_n^{\alpha,\beta}$ selects every $(2^\alpha)^{\text{th}}$ point in the vertical direction and every $(2^\beta)^{\text{th}}$ point in the horizontal direction of $\mathbb{Z}_n^2$. Each normal point $w \in \mathbb{W}_n^{\alpha,\beta}$ corresponds to a $2^\alpha \times 2^\beta$ *normal rectangle*, whose north-west corner is w. We will also use the notation $\mathbb{W}_n^{\alpha,\beta}$ to denote the set of normal rectangles. Let $\mathbb{W}_n = \bigcup_{\alpha=0}^{n} \bigcup_{\beta=0}^{n} \mathbb{W}_n^{\alpha,\beta}$ denote the set of all possible normal rectangles (and respective normal points). The figure above shows a $(1, 2)$-normal subset and the respective $2^1 \times 2^2$ normal rectangles $\mathbb{W}_4^{1,2}$ of $\mathbb{Z}_4^2$.

Lemma 2 *Any $a \times b$ rectangle $x \in \mathbb{Z}_n^2$, contains an $a' \times b'$ normal rectangle $x' \subseteq x$ with $1 \leq |x|/|x'| \leq 16$, such that $1 \leq a/a' \leq 4$ and $1 \leq b/b' \leq 4$.*

Given a rectangle x, the internal normal rectangle x' of Lemma 2 can be computed in constant time. See the figure above for an example rectangle x and its respective internal normalized rectangle x'.

Aspect Ratios. The aspect ratio of an $a \times b$ rectangle is $\frac{b}{a}$. Each normal rectangle induced by $\mathbb{W}_n^{\alpha,\beta}$ has an aspect ratio of $2^{\beta-\alpha}$. All rectangles induced by $\mathbb{W}_n^{\alpha+i,\beta+i}$ (for integer i) have aspect ratio $2^{\beta-\alpha}$. Thus, given g, $0 \le g \le n$, $\mathbb{W}_n^{\alpha,g+\alpha}$ corresponds to normal rectangles of aspect ratio 2^g and size $2^\alpha \times 2^{g+\alpha}$. Let $\mathbb{W}_n^{(g,+)} = \bigcup_{\alpha=0}^{n-g} \mathbb{W}_n^{\alpha,g+\alpha}$ denote the normal rectangles of aspect ratio 2^g. Similarly, let $\mathbb{W}_n^{(g,-)} = \bigcup_{\alpha=0}^{n-g} \mathbb{W}_n^{g+\alpha,\alpha}$ denote the normal rectangles of aspect ratio 2^{-g}. Let $\mathbb{W}_n^{(+)} = \bigcup_{g=0}^{n} \mathbb{W}_n^{(g,+)}$ be the set of all possible normal rectangles with aspect ratio at least one. Let $\mathbb{W}_n^{(-)} = \bigcup_{g=1}^{n} \mathbb{W}_n^{(g,-)}$ be the set of all possible normal rectangles with aspect ratio < 1.

Union of Rectangles. In our KMP approach, given a rectangle x_i we compute inside it a y_i which is either a normal rectangle (for example given by Lemma 2), or y_i is a rectangle that consists of multiple normal rectangles (as will be the case of Section 5). Consider a sequence of rectangles $x_0, x_1, \ldots, x_{m-1}$. Denote the union of these rectangles as $X = x_0 \cup x_1 \cup \cdots \cup x_{m-1}$ and the area of X as $|X|$. Consider also a sequence of rectangles $y_0, y_1, \ldots, y_{m-1}$, where each y_i is contained in x_i, with union $Y = y_0 \cup y_1 \cup \cdots \cup y_{m-1}$. We are interested in estimating the area of X based on calculating the area of Y.



Let x be an $a \times b$ rectangle (see the figure above) with $a \cdot b$ discrete points in $\mathbb{Z}_n^2$, $0 \le a, b < 2^n$. Denote by $|x|$ the area of x, namely, $|x| = a \cdot b$. Let y denote an $a' \times b'$ rectangle in x. Write $a = a^{top} + a' + a^{bottom}$ and $b = b^{left} + b' + b^{right}$, where $a = a' c_{x,v}$ and $b = b' c_{x,h}$, for some integers $c_{x,v}, c_{x,h} \ge 1$. Note that $|x| = a' c_{x,v} \cdot b' c_{x,h} = c_x |y|$, where $c_x = c_{x,v} \cdot c_{x,h}$.

We define rectangle z^{top} of dimensions $a^{top} \times b'$ that resides on top of y. Similarly, we define z^{bottom} as an $a^{bottom} \times b'$ rectangle that resides on the bottom of y. Symmetrically, we define z^{left} and z^{right} as the $a' \times b^{left}$ and $a' \times b^{right}$ rectangles that reside on the left and right of y. Note that $z^{top}, z^{bottom}, z^{left}$, and z^{right} are all in x. Finally, we define the *cross* polygon z to be: $z = y \cup z^{top} \cup z^{bottom} \cup z^{left} \cup z^{right}$.

Given X and Y, we define the corresponding sequence of cross polygons $z_0, z_1, \ldots, z_{m-1}$, with union $Z = z_0 \cup z_1 \cup \cdots \cup z_{m-1}$. Denote $c_v = \max_i c_{x_i,v}$, and $c_h = \max_i c_{x_i,h}$. It can be shown that $|Z| = \alpha |Y|$, for $1 \le \alpha \le 2c_v + 2c_h - 3$. It can also be shown that $|X - Z| = \beta |Z|$, for $0 \le \beta \le 2 \cdot \min\{c_v, c_h\} - 2$. Therefore, we obtain:

Lemma 3 *$|X| = \gamma |Y|$, where $1 \le \gamma \le (2c_v + 2c_h - 3)(2 \cdot \min\{c_v, c_h\} - 1)$.*

4 Proximity Transformation

Here we describe the Proximity transformation which deterministically maps each normal rectangle to a one-dimensional range based on a Z-ordering.

Without loss of generality consider a bucket $\mathbb{B}_g$ which contains normal rectangles from $\mathbb{W}_n^{(g,-)}$. For the one-dimensional mapping (step 5 of Algorithm 1), we show how Algorithm Proximity takes a normalized rectangle $x' \in \mathbb{B}_g$ and maps it to a linear interval (range) r in a manner that preserves the intersection properties of the rectangles of $\mathbb{B}_g$.

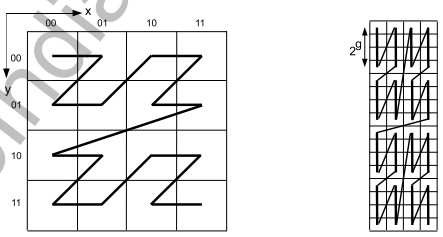


Figure 1: A Z-curve for aspect ratio (a) 1, (b) 1/4

Define $f_n^{(0,+)} : \mathbb{Z}_n^2 \longrightarrow \mathbb{Z}_{2n}$ as the well-known Z-ordering [9] (see Fig. 1a). The figure shows the Z-values for the case $g = 0$ of points for the two dimensional space $0 \le x \le 3, 0 \le y \le 3$ (shown in binary). Interleaving the binary coordinate values gives binary Z-values and connecting the Z-values in their order from the lowest (0000) to the highest (1111) produces the recursively Z-shaped curve, i.e., a Z-order. For aspect ratio < 1, define $f_n^{(g,-)} : \mathbb{Z}_n^2 \longrightarrow \mathbb{Z}_{2n}$ as follows. Partition $\mathbb{Z}_n^2$ into 2^g contiguous vertically aligned elements, then connect them in the order provided by a Z-ordering (see Fig. 1b). For aspect ratio > 1, partition as horizontally aligned 2^g elements. Note that $f_n^{(g,-)}$ preserves the intersection properties of normalized rectangles of $\mathbb{W}_n^{(g,-)}$, that is, for any $x'_1, x'_2 \in \mathbb{W}_n^{(g,-)}$, $|x'_1 \cap x'_2| = |(f_n^{(g,-)}(x'_1)) \cap (f_n^{(g,-)}(x'_2))|$. Similarly, $f_n^{(g,+)}$ preserves the intersection properties of normalized rectangles in $\mathbb{W}_n^{(g,+)}$.

Each element i of $\mathbb{Z}_{2n} = \{0, 1, \cdots, 2^{2n} - 1\}$ (the codomain of $f_n^{(g,-)}(f_n^{(g,+)})$) is a $2n$-bit binary representation of i. The domain $\mathbb{Z}_n^2$ of $f_n^{(g,-)}$ ($f_n^{(g,+)}$) consists of doublets (p, q), where $p, q \in \mathbb{Z}_n$. So, each (p, q) expressed as the concatenation of the binary representa-

tions of p and q is a $2n$-bit quantity. This $2n$-bit number (belonging to $\mathbb{Z}_{2n}$) is called the *row-major index* of (p, q). Thus, function $f_n^{(g,-)}$ ($f_n^{(g,+)}$) can be viewed as the transformation of one $2n$-bit number into another $2n$-bit number.

Since $n = \mathcal{O}\left(\log|\mathbb{Z}_n^2|\right)$, the function $f_n^{(g,-)}(\cdot)$ ($f_n^{(g,+)}(\cdot)$) can be computed using a logarithmic-size (albeit non-standard) operation that could be assumed to be computable in constant time. Therefore, we obtain:

Lemma 4 *For any normalized rectangle $x \in \mathbb{B}_g$, $f_n^{(g,-)}(x)$ ($f_n^{(g,+)}(x)$) is a set of $|x|$ contiguous integers (range) in $\mathbb{Z}_{2n}$. This ordering can be computed in constant time for a given rectangle and preserves the intersection properties of rectangles of $\mathbb{B}_g$.*

5 One Bucket

We describe how to efficiently estimate the area of a stream with fat rectangles. We first describe the special case of a stream of squares $\Upsilon = \{x_0, x_1, \ldots, x_{m-1}\}$ and then we extend the result to include fat rectangles with aspect ratios other than 1. The normalization (step 4) of Algorithm 1 uses only one bucket $\mathbb{B}$ which contains normal rectangles of aspect ratio 1 (namely, normal squares in $\mathbb{W}_n^{(0,+)}$). Using the following result it is possible to transform each square $x \in \Upsilon$ into a set of normal squares which are then used to estimate the total area A of the stream Υ.

Lemma 5 *Given an $a \times a$ square x and η, $0 < \eta \leq 1$, there is an internal $a' \times a'$ square x' which consists of at most c/η normal squares (of possibly various sizes), for some positive constant c, such that $a(1-\eta) \leq a' \leq a$.*

In Algorithm 1, since $\text{est}(A) = \sqrt{E_{\max} E_{\text{sum}}}$ and we use only one bucket, we get $E_{\max} = E_{\text{sum}}$. Thus, the result of the algorithm is directly the output of Hits (Theorem 1) on bucket $\mathbb{B}$. For each square $x_i \in \Upsilon$, let x_i' be the respective square given by Lemma 5. Suppose that the set $\Upsilon' = \{x_0', x_1', \ldots, x_{m-1}'\}$ has total area A'. Each x_i is replaced with c/η normal squares that cover the respective x_i' (as specified by Lemma 5), and further each such normal square is projected to a range through the Proximity transformation of Section 4. In other words, Hits returns an (ϵ', δ) estimate on A', for $0 < \epsilon', \delta < 1$. This is then used to obtain an estimate of A. Using Lemma 3 we can relate the areas of A and A' through η, since $c_v, c_h \leq 1/(1-\eta)$. Finally, by appropriately substituting ϵ' and η with linear functions of ϵ, we obtain an (ϵ, δ)-approximation for $\text{est}(A)$. The space complexity and query time remain asymptotically the same as in Hits, while the time complexity increases by a factor of c/η (due to the number of normalized rectangles inside each x_i'). Therefore, we can obtain:

Theorem 6 *For the special case of a stream Υ of squares, given $0 < \epsilon, \delta < 1$, Algorithm 1 (ϵ, δ)-approximates the area of A with space complexity $\mathcal{O}(\frac{1}{\epsilon^2} \log U \log \frac{1}{\delta})$ bits, amortized time taken to process a square $\mathcal{O}(\frac{1}{\epsilon} \log \frac{U}{\epsilon} \log \frac{1}{\delta})$, and time taken to process a query for A at any time $\mathcal{O}(1)$.*

Theorem 6 can be also applied to fat rectangles. Each fat rectangle of aspect ratio h can be converted to a stream of at most $h+1$ (or $1/(h+1)$ if $h < 1$) squares that cover the rectangle area. Thus, we can use the approach above for a constant factor increase in time.

6 Many Buckets

Here we give an analysis of Algorithm 1 for the general case of arbitrary input rectangles. Section 6.1 establishes a relation of (ϵ, δ)-estimators to ξ-approximations.

6.1 Area Estimation

Consider a set $\Upsilon = \{x_0, x_1, \ldots, x_{m-1}\}$ of m rectangles in $\mathbb{Z}_n^2$, such that each x_j is an $a_j \times b_j$ rectangle, $0 \leq a_j, b_j < \sqrt{U}$, with $a_j b_j$ discrete points, and total area A. Partition Υ arbitrarily into χ subsets $\{\Upsilon_0, \Upsilon_1, \ldots, \Upsilon_{\chi-1}\}$ where Υ_i has m_i rectangles and $m = \sum_{i=0}^{\chi-1} m_i$. Let A_i denote the area of the union of the rectangles in Υ_i. In each x_j of size $a_j \times b_j$, let x_j' denote an $a_j' \times b_j'$ rectangle contained in x_j, where $a_j' \leq a_j$ and $b_j' \leq b_j$ Define $\Upsilon', \Upsilon_i', A'$, and A_i' correspondingly. Let $0 < \Delta_1 \leq \frac{A'}{A} \leq \Delta_2 \leq 1$; we fix Δ_1 and Δ_2 later in Section 6.2.

Let us assume that there is an (ϵ, δ)-estimator algorithm $\mathcal{A}$ (similar to Theorem 1) that computes an estimate $\text{est}(A_i')$ of the area in the rectangles in Υ_i' and returns: (i) $E_{\max} = \max_i \text{est}(A_i')$, the maximum among $\text{est}(A_i'), 0 \leq i \leq \chi - 1$, and (ii) $E_{\text{sum}} = \sum_{i=0}^{\chi-1} \text{est}(A_i')$, the sum of each $\text{est}(A_i'), 0 \leq i \leq \chi - 1$. The algorithm $\mathcal{A}$ then uses the estimates $E_{\max}$ and E_{sum} to estimate quantity $\text{est}(A)$ of A as $\text{est}(A) = \sqrt{E_{\max} E_{\text{sum}}}$. Define relative error as $\varrho = \frac{|\text{est}(A) - A|}{A}$. We have that:

Lemma 7 *The (ϵ, δ)-estimator algorithm $\mathcal{A}$ computes an estimate $\text{est}(A)$ of the total area A such that $A \cdot \epsilon_1 \leq \text{est}(A) \leq A \cdot \epsilon_2$ with probability $(1-\delta)^\chi$, where χ is the number of blocks in a partition of set Υ of input rectangles, $\epsilon_1 = \frac{\Delta_1(1-\epsilon)}{\sqrt{\chi}}$, $\epsilon_2 = \Delta_2(1+\epsilon)\sqrt{\chi}$, and $0 < \Delta_1 \leq \Delta_2 \leq 1$.*

6.2 Approximation

We project each bucket $\mathbb{B}_i$ to $\mathbb{Z}_{2n}$ (as described in Section 4) and apply a range-efficient algorithm for F_0. Let Υ_i denote the original rectangles in Υ that are projected with bucket $\mathbb{B}_i$, and let Υ_i' denote the normal rectangles assigned to $\mathbb{B}_i$ (according to their aspect ratio).

Thus, $\Upsilon = \{\Upsilon_0, \Upsilon_1, \ldots, \Upsilon_{\chi-1}\}$, and after normalization $\Upsilon' = \{\Upsilon'_0, \Upsilon'_1, \ldots, \Upsilon'_{\chi-1}\}$. For each $x \in \Upsilon$ we use a single internal normal rectangle x' as given by Lemma 2, which is assigned to a bucket $\mathbb{B}_i$ according to its aspect ratio. Let A_i (resp. A'_i) denote the total area of Υ_i (resp. Υ'_i) . This results to $A_i/A'_i = \gamma$, $\gamma \leq 91$, from the area analysis in Lemma 3.

The Hits algorithm (Theorem 1) yields an (ϵ, δ)-approximation of F_0 for the ranges in each $\mathbb{B}_i$. The estimate $\text{est}(A)$ of the total area A of Υ using $E_{\max}$ and E_{sum} is computed from the (ϵ, δ)-estimates of A'_i given by the Hits algorithm for each bucket.

From Lemma 7, substituting $\Delta_1 = 1/\gamma$ and $\Delta_2 = 1$, Algorithm 1 computes an estimate $\text{est}(A)$ of the total area A of the set Υ of m rectangles mapped to χ buckets such that $A \cdot \epsilon_1 \leq \text{est}(A) \leq A \cdot \epsilon_2$, where $\epsilon_1 = \frac{1}{\gamma}\frac{(1-\epsilon)}{\sqrt{\chi}}$ and $\epsilon_2 = (1+\epsilon)\sqrt{\chi}$ with probability $(1-\delta)^{\chi}$. We set $\chi = 2n + 1 = \log U + 1$ (the total number of normal aspect ratios), and hence, $\epsilon_1 = \frac{1}{\gamma}\frac{(1-\epsilon)}{\sqrt{\log U+1}}$ and $\epsilon_2 = (1+\epsilon)\sqrt{\log U + 1}$ with probability $(1-\delta)^{\log U+1}$.

Algorithm 1 reaches $\text{est}(A)$ with space complexity $\mathcal{O}(\frac{1}{\epsilon^2}\log^2 U \log\frac{1}{\delta})$ bits, amortized time taken to process a rectangle $\mathcal{O}(\log\frac{U}{\epsilon}\log\frac{1}{\delta})$, and the time taken to process a query for F_0 is $\mathcal{O}(\log U)$. These follow from Theorem 1 by combining the space and time complexities of $\log U + 1$ instances of Hits. In comparison to Hits, the space needed by our approximation algorithms increases by a factor of $\mathcal{O}(\log U)$ due to the $\log U + 1$ buckets. The amortized time taken to process a rectangle remains the same as we need to run Hits on only a single bucket for that rectangle. Since it is necessary to compute the median of $\log U + 1$ instances of Hits, one for each bucket, the time taken to process a query for A increases by a factor of $\mathcal{O}(\log U)$. By setting $\epsilon = 1/2$ and $\delta = \frac{1}{\log U+1}$, our algorithm approximates $\text{est}(A)$, such that $\Omega(1/\sqrt{\log U}) \leq \text{est}(A)/A \leq \mathcal{O}(\sqrt{\log U})$, with constant probability.

Theorem 8 *Algorithm 1 $\mathcal{O}(\sqrt{\log U})$-approximates A with constant probability and achieves space complexity $\mathcal{O}(\log^2 U \cdot \log\log U)$ bits, amortized time to process a rectangle $\mathcal{O}(\log U \cdot \log\log U)$, and time taken to process a query at any time $\mathcal{O}(\log U)$.*

The asymptotic notation $\mathcal{O}(\sqrt{\log U})$ hides large constants, due to the fact that for each x_i we use some internal normalized square. We can improve the constants and bring them down to 1, by tiling each x_i with poly-log number of normalized rectangles as specified by the lemma below. This has the side effect of increasing the time complexity by a poly-log factor which can be traded-off for the enhanced accuracy.

Lemma 9 *A rectangle r can be $(1-\eta)$-approximately tiled with $4\log^2\frac{1}{\eta^2}$ normal rectangles, for $0 < \eta \leq 1/2$.*

7 Conclusions

We presented a randomized approximation algorithm with poly-log bounds on time and space complexity with approximation factor $1 \pm \epsilon$ for fat rectangles and $\mathcal{O}(\sqrt{\log U})$ for general rectangles for the KMP in the streaming model. Our technique will give an $\mathcal{O}((\sqrt{\log U})^{d-1})$-approximation for the general case of d-dimensional KMP. For future work, it would be interesting to explore techniques that will reduce the current approximation factor $\mathcal{O}(\sqrt{\log U})$ for general rectangles to $\mathcal{O}(1)$ or $1 \pm \epsilon$, using deterministic transformations. Moreover, it would be interesting to evaluate our algorithm experimentally in a real-time setting.

Acknowledgements: We are indebted to Srikanta Tirthapura for helpful suggestions for the problem.

References

[1] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *STOC*, pages 20–29, 1996.

[2] M. Benedikt and L. Libkin. Exact and approximate aggregation in constraint query languages. In *PODS*, pages 102–113, 1999.

[3] J. Bentley. Algorithms for Klee's rectangle problems, Unpublished notes, Computer Science Department, Carnegie Mellon University. 1978.

[4] K. Bringmann. An improved algorithm for Klee's measure problem on fat boxes. *Comput. Geom. Theory Appl.*, 45(5-6):225–233, 2012.

[5] K. Bringmann and T. Friedrich. Approximating the volume of unions and intersections of high-dimensional geometric objects. *Comput. Geom. Theory Appl.*, 43(6-7):601–610, 2010.

[6] E. Y. Chen and T. M. Chan. Space-efficient algorithms for Klee's measure problem. In *CCCG*, pages 27–30, 2005.

[7] V. Klee. Can the measure of $\cup[a_i, b_i]$ be computed in less than $O(n \log n)$ steps? *American Mathematical Monthly*, 84(4):284–285, 1977.

[8] G. Kuper, L. Libkin, and J. Paredaens. *Constraint Databases*. Springer, 1st edition, 2010.

[9] J. A. Orenstein and T. H. Merrett. A class of data structures for associative searching. In *PODS*, pages 181–190, 1984.

[10] A. Pavan and S. Tirthapura. Range-efficient counting of distinct elements in a massive data stream. *SIAM J. Comput.*, 37(2):359–379, 2007.

[11] S. Tirthapura and D. P. Woodruff. Rectangle-efficient aggregation in spatial data streams. In *PODS*, pages 283–294, 2012.

[12] J. Vahrenhold. An in-place algorithm for Klee's measure problem in two dimensions. *Inf. Process. Lett.*, 102(4):169–174, 2007.